COMSC-210 Lab 13
O(n-squared) Sorting Algorithms

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this assignment you will implement n-squared sorting in one of your existing templates.

After you complete this lab assignment, post the required files to the COMSC server so that you receive the proper credit for this 50-point lab. Your posted files are due at midnight of the evening of the due date indicated in the course outline. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab13** folder provided.

LAB 13a: Write A Sortable Array Class Template `[ SortableStaticArray.h and MySortableStaticArray.cpp ]`
Purpose. The purpose of this lab is to implement built-in sorting in an array template. The resulting template can be used in any program in place of a C++ array whose data type supports operator-equals-equals.

Requirements. Rewrite `StaticArray.h` and `MyStaticArray.cpp` from lab 3a as **`SortableStaticArray.h`** and **`MySortableStaticArray.cpp`**, adding a sort function. Name the new templated class `SortableStaticArray`. Do NOT use inheritance -- do copy/paste/mark-up.

Use this prototype for a **public** sort function:

```
void sort();
```

...implementing either nested for-loop sorting or insertion sort from the lecture notes -- *your choice*. Note that the array *may have holes*, so account for that.

Improve the app so that after all data entry is complete, it:

1. output the data structure *size* after Q was entered, exactly as in lab 3b,
2. output the list of all entered values with their indexes, exactly as in lab 3b,
3. implement an event-controlled loop that prompts for an index value and outputs whether the index is in use or not, and if in use, what is the value stored for that index, exactly as in lab 3b, Allowing multiple searches until the user elects to stop by entering uppercase or lowercase Q, and.
4. **sorts the data entries made by the user, lo-to-hi for the entered values, and outputs them in pairs of indexes and values**.

Here's a sample of how this should work (user input in blue):

```
Input an index and a value [Q to quit]: 33 12
Input an index and a value [Q to quit]: 4 100
Input an index and a value [Q to quit]: 5 300
Input an index and a value [Q to quit]: x 17
Input an index and a value [Q to quit]: 33 120
Input an index and a value [Q to quit]: -1 234
```

```
Input an index and a value [Q to quit]: q

I stored this many values: 4
The values are:
  0 17
  4 100
  5 300
  33 120

Input an index for me to look up [Q to quit]: 33
Found it -- the value stored at 33 is 120
Input an index for me to look up [Q to quit]: 1000
I didn't find it
Input an index for me to look up [Q to quit]: Q

The sorted values are:
  0 17
  4 100
  5 120
  33 300
```

Submit the two files (1 CPP and 1 H) to the class website for credit.

Program I/O. Input: Same as lab 3b. Output: Same as lab 3b, plus sorted results.

LAB 13b: Applying A Data Structure To A Database Program `[ DvcSchedule13b.cpp ]`
Purpose. In this lab you will apply a sort function to a database program.

Requirements. Rewrite `DvcSchedule6b.cpp` from lab 6b as **`DvcSchedule13b.cpp`**, but use your new SortableStaticArray template for SUBJECT objects, use SortableStaticArray. Then instead of using the nested-for-loop sorting, call SortableStaticArray::sort

Submit the CPP to the class website for credit. Do NOT submit the TXT input file.

Program I/O. Same as lab 6b.

How to pause a console program: *show / hide*

GRADING RUBRIC *show/hide*
